Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software UPM ETSIINF.
Examen de Programación II. Convocatoria ordinaria. 27-4-2016.

Realización: *El test se realizará en la hoja de respuesta. Es **importante** que no olvidéis rellenar vuestros datos personales y el código clave de vuestro enunciado. Se pueden utilizar hojas aparte en sucio.*

Duración: *La duración total del test será de **50 minutos**.*

Puntuación: *El test se valora sobre **10 puntos**. Las preguntas tipo test pueden tener una única respuesta o varias respuestas, el enunciado lo deja claro. Cada pregunta con una única respuesta respondida correctamente vale 1 punto, e incorrectamente respondida resta 1/3 puntos. Si en una pregunta con una única respuesta se selecciona más de una respuesta, la pregunta se puntuará con 0 puntos. Para una pregunta con varias respuestas, cada afirmación correcta seleccionada suma 1/no_respuestas_correctas puntos, y cada afirmación incorrecta seleccionada resta 1/no_respuestas_incorrectas puntos. Las preguntas no contestadas suman 0 puntos en cualquier caso.*

Calificaciones: *Las calificaciones se publicarán en moodle como muy tarde el día **3 de Mayo de 2016***

Revisión: *Las revisiones serán el día **5 de Mayo de 2016** previa petición por correo electrónico al profesor que se indique por el foro de la asignatura.*

Primer Ejercicio

Dada la siguiente clase *Libro*:

```
public class Libro {
   private String name;
   private boolean prestado = false;

   public Libro(String name,
                boolean prestado){
      this.name = name;
      this.prestado = prestado;
   }
   public Libro(Libro libro){
      this.name = libro.name;
   }
   public String toString(){
      return this.name + ":"+this.prestado;
   }
   public boolean esIgual(Libro libro){
      return this.name.equals(libro.name) &&
             this.prestado == libro.prestado;
   }
}
```

y dado el siguiente *main*():

```
public static void main(String[] args) {
   Libro libro1 = new Libro("Todo Java",true);
   Libro libro2 = new Libro(libro1);
   System.out.println(libro1 + "**" + libro2+
        "**"+libro1.esIgual(libro2));
}
```

Pregunta 1 (1 punto)

Indica cuál será la salida por consola tras ejecutar el *main*() anterior (**sólo una**).

a) Todo Java:true**Todo Java:false**false
b) Todo Java:false**Todo Java:false**true
c) Todo Java:false**Todo Java:true**false
d) Todo Java:true**Todo Java:true**true

Segundo Ejercicio

Dada las siguientes clases *Fecha* y *Persona*:

```
public class Fecha {
 private int dia, mes, anio;

 public boolean esIgual(Fecha fecha){
    return (dia == fecha.dia) &&
           (mes == fecha.mes) &&
           (anio == fecha.anio);
 }
 // ...
}

public class Persona {
 private int dni;
 private String nombre;
 private Fecha fecha;

 public boolean esIgual(Persona persona){
    //TODO
 }
 // ...
}
```

Pregunta 1 (1 punto)

Indica cuál de las siguientes opciones debería aparecer en el cuerpo del método esIgual (**sólo una**).

a) return dni==persona.dni &&
nombre.equals(persona.nombre) &&
fecha.esIgual(persona.fecha);
b) return dni==persona.dni &&
nombre == persona.nombre &&
fecha.esIgual(persona.fecha);
c) return dni==persona.dni &&
nombre == persona.nombre &&
fecha == persona.fecha;
d) return dni==persona.dni &&
nombre.equals(persona.nombre) &&
fecha == persona.fecha;

Continúa...

Tercer Ejercicio

Teniendo en cuenta el mecanismo de paquetes de Java.

Pregunta 1 (1 punto)

Señala todas las afirmaciones verdaderas. **Puede haber más de una respuesta correcta**.

a) NO puede haber clases que se llamen igual en paquetes distintos

b) Puede haber clases que se llamen igual en paquetes distintos

c) Para poder usar código de una clase A en otra clase B que está en OTRO paquete hace falta importar la clase A en la clase B

d) Para poder usar código de una clase A en otra clase B que está en el MISMO paquete hace falta importar la clase A en la clase B

Cuarto Ejercicio

Dada la siguiente clase *Numeros*:

```
public class Numeros {
  private int[] valores;
  private int numValores;

  public Numeros(int capacidad){
     this.numValores = capacidad;
  }

  public int metodoA(int numero){
     int resultado = 0;
     for(int i = 0; i < this.numValores; i++)
         resultado += valores[i]+numero;
     return resultado;
  }
}
```

y dado el siguiente *main*():

```
public static void main(String[] args) {
    Numeros secuencia = new Numeros(10);
    System.out.println(secuencia.metodoA(2));
}
```

Pregunta 1 (1 punto)

Indica cuál será la salida por consola tras ejecutar el *main*() anterior (**sólo una**).

a) 20

b) 10

c) 0

d) Exception in thread "main"
java.lang.NullPointerException

Quinto Ejercicio

En relación al tema de los niveles de visibilidad.

Pregunta 1 (1 punto)

Señala todas las afirmaciones verdaderas. **Puede haber más de una respuesta correcta**.

a) Los métodos auxiliares deben ser privados.

b) Los métodos auxiliares deben ser SIEMPRE públicos para que los puedan usar objetos de otras clases.

c) Los servicios que ofrece una clase se deben declarar como métodos privados.

d) Para evitar problemas los atributos siempre deben ser privados excepto cuando hay herencia de clases que pueden ser también protected.

Sexto Ejercicio

Dado el siguiente fragmento de código que utiliza las clases `NodoEntero` y `CadenaEnlazada` vistas en clase:

```
public static void main(String[] args) {

 CadenaEnlazada cadena = new CadenaEnlazada();

 NodoEntero nodo1 = new NodoEntero(1, null);
 NodoEntero nodo2 = new NodoEntero(2, null);
 NodoEntero nodo3 = new NodoEntero(3, null);

 nodo3.setSiguiente(nodo2);
 cadena.setCabeza(nodo3);
 nodo2.setSiguiente(nodo1);

 NodoEntero nodo = cadena.getCabeza();
 while (nodo.getSiguiente() != null) {
    System.out.print(nodo.getDato() + " - - ");
    nodo = nodo.getSiguiente();
 }
 System.out.println("Se acabo");
}
```

Pregunta 1 (1 punto)

Indica cuál de las siguientes opciones es la verdadera para la salida de la consola (**sólo una**)

a) 3 - 2 - 1 - Se acabo

b) Exception in thread "main"
java.lang.NullPointerException

c) 1 - 2 - 3 - Se acabo

d) 3 - 2 - Se acabo

Continúa...

Séptimo Ejercicio

Dada la siguiente clase *Concierto* y suponiendo ya implementada una clase *Cliente*:

```
public class Concierto {

  private Cliente[] clientesEsperando;
  private int noClientesEsperando;
  private int entradasDisponibles;

  public Concierto(int capacidadMax){
     entradasDisponibles = capacidadMax;
     noClientesEsperando = 0;
     clientesEsperando =
             new Cliente[capacidadMax];
  }

  public void llegaAVentanilla(Cliente cliente){
     // si no quedan entradas, el cliente
     // no se pone en la cola y se marcha
     if (entradasDisponibles > 0){
        // el cliente se pone en la cola de
        // la ventanilla
        clientesEsperando[noClientesEsperando] = cliente;
        noClientesEsperando++
        entradasDisponibles--;
     }
  }

  public Cliente atiendeCliente(){
     Cliente clienteAtendido = clientesEsperando[0];
     for(int i=0; i<noClientesEsperando-1; i++)
        clientesEsperando[i] = clientesEsperando[i+1];

     clientesEsperando[noClientesEsperando-1] = null;
     noClientesEsperando--;
     return clienteAtendido;
  }
 }
```

y dado el siguiente main(), indica cuál será el contenido final del vector de clientes en el objeto concierto1 tras ejecutar este *main*():

```
public static void main(String[] args) {

    Cliente c1, c2, c3, c4, c5, c6;

    c1 = new Cliente(); c2 = new Cliente();
    c3 = new Cliente(); c4 = new Cliente();
    c5 = new Cliente(); c6 = new Cliente();

    Concierto concierto1 = new Concierto(3);

    concierto1.llegaAVentanilla(c1);
    concierto1.llegaAVentanilla(c2);
    concierto1.llegaAVentanilla(c3);
    concierto1.llegaAVentanilla(c4);

    concierto1.atiendeCliente();

    concierto1.llegaAVentanilla(c5);
    concierto1.llegaAVentanilla(c6);
}
```

Pregunta 1 (1 punto)

Indica cuál de las siguientes afirmaciones es verdadera (**sólo una**) en donde el primer elemento de la tupla ocupa la posición 0 y el último la posición 2

a) (c2, c3, c4)

b) (c2, c3, null)

c) (c1, c2, null)

d) (c4, c5, c6)

Continúa...

Octavo Ejercicio

Dado el siguiente fragmento de código que debería ordenar de menor a mayor las primeras `numElementos` posiciones de la variable `vector`:

```
public static void main(String[] args) {
  int[] vector = ....;
  int numElementos = ....;
  // Bucle de elementos a insertar
  for (int k = 1; k < numElementos; k++) {
    int aux = vector[k]; // elem a insertar
    // Bucle de mover los mayores ya
    // ordenados
    int pos = k;
    while (pos >= 0 && vector[pos - 1] > aux){
        vector[pos] = vector[pos - 1];
        pos--;
    }
    vector[pos] = aux;
  }
}
```

Pregunta 1 (1 punto)

Este código contiene un defecto que impide obtener un resultado correcto. Indica cuál de las siguientes opciones señala el defecto que hay en el código (**sólo una**)

a) $vector[pos-1] > aux$ debería ser $vector[pos] > aux$

b) $int\ k = 1$ debería ser $int\ k = 0$

c) $k < numElementos$ debería ser $k < numElementos - 1$

d) $pos >= 0$ debería ser $pos > 0$

Noveno Ejercicio

Dado el siguiente fragmento de código que maneja una cadena basada en la implementación de `NodoEntero` utilizada en la asignatura:

```
NodoEntero cadena = new NodoEntero(4, null);
cadena.setSiguiente(new NodoEntero(3, null));
cadena.getSiguiente().setSiguiente(new NodoEntero(2, null));
NodoEntero aux = cadena.getSiguiente();
aux.setSiguiente(aux.getSiguiente().getSiguiente());
```

Pregunta 1 (1 punto)

Indicar cuál es el contenido de la cadena resultante (**sólo una**).

a) 4, 2

b) 2, 3, 4

c) 4, 2, 3

d) 4, 3

Décimo Ejercicio

Dadas las clases `Punto` y `Circulo`:

```
public class Punto {
    private double x,y;
    public Punto(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public Punto(Punto p){
        this.x = p.x;
        this.y = p.y;
    }
    public void setX(int x){
        this.x = x;
    }
    public String toString(){
        return x + "," + y;
    }
}
public class Circulo {
    private Punto centro;
    private double radio;
    public Circulo(Punto centro,
                   double radio) {
        this.centro = centro;
        this.radio = radio;
    }
    public String toString(){
        return "C:" + centro
            + " R:" + radio;
    }
}
```

Y dado el siguiente programa principal:

```
public class TestCirculo {
  public static void main(String[] args) {
    Punto centro = new Punto(1,2);
    Circulo circulo1 = new Circulo(centro, 7);
    centro.setX(5);
    // centro.setX(1);
    System.out.println(centro + " "
       + circulo1);
  }
}
```

La salida correcta de este main debería ser 5.0,2.0 C:1.0,2.0 R:7.0, pero cabe la posibilidad de que no lo sea.

Pregunta 1 (1 punto)

Indica cuál es la salida por consola que se produce al ejecutar este main y qué habría que cambiar en el código para que la salida sea la que se indica como correcta (**sólo una**):

a) la salida es 1.0,2.0 C:1.0,2.0 R:7.0 y habría que inicializar `centro` en el constructor así: `this.centro = new Punto(centro);`

b) la salida es 5.0,2.0 C:5.0,2.0 R:7.0 y habría que descomentar la línea del main `centro.setX(1);`

c) No hay que cambiar nada porque la salida ya es correcta

d) la salida es 5.0,2.0 C:5.0,2.0 R:7.0 y habría que inicializar `centro` en el constructor así: `this.centro = new Punto(centro);`

Continúa...